"Express Mail" Mailing Label No. **EL917901808US**

PATENT APPLICATION
ATTORNEY DOCKET NO. SUN-P6407-PIP

METHOD AND APPARATUS FOR PARTITIONING RESOURCES WITHIN A COMPUTER SYSTEM

Inventors: Stephen C. Hahn and Tim P. Marsland

BACKGROUND

Field of the Invention

The present invention relates to operating systems for computers. More specifically, the present invention relates to a method and an apparatus for allocating computer system resources between different concurrently executing workloads.

Related Art

The advent of computer networks has led to the development of server computer systems that perform computational operations on behalf of numerous

client computer systems. These server computer systems are typically configured with large amounts of computing resources, such as processors and memory, and are typically employed in processing one or more concurrently executing computational workloads.

One challenge in designing an operating system to manage such a server is to ensure that computer system resources are allocated between computational workloads so that the minimum requirements of each workload are satisfied, and so that the workloads are collectively executed in an efficient manner.

Some modern computing systems provide support for partitioning a machine-wide resource into smaller sets and then associating one or more workloads with each of the sets. For example, the SOLARIS™ operating system, distributed by SUN Microsystems, Inc. of Palo Alto, California, allows processors to be grouped into processor sets, wherein specific processes may be bound to a specific processor set. In this way, the specific processes do not compete with other processes for access to the specific processor set.

However, these partitioning operations must presently be specified manually by a machine operator and are dependent upon the specific machine configuration, as well as the operator's awareness of resource requirements for excepted workloads. Furthermore, a given allocation of computer system resources is not persistent across machine failures.

Other operating systems have developed a mechanism for assembling a group of resources into a fixed "container" that processes can bind to in order to access the resources. However, resources within a fixed container cannot be flexibly changed over time to accommodate changing resource requirements for the various system workloads. Furthermore, resources cannot be shared between containers.

What is needed is a method and an apparatus for allocating computer system resources between different concurrently executing workloads without the above-described problems and without other limitations which may become apparent upon reading this specification.

SUMMARY

One embodiment of the present invention provides a system that allocates computer system resources between concurrently executing workloads. The system operates by establishing a first resource pool that specifies requirements for different computer system resources. Next, the system allocates the different computer system resources to one or more resource pools, including the first resource pool, to create a resource allocation, wherein requirements of the first resource pool are satisfied, and wherein resources allocated to the first resource pool can change over time. The system then binds a first process to the first resource pool, so that the first process has access to the plurality of different computer system resources allocated to the first resource pool.

In one embodiment of the present invention, while allocating different computer system resources, the system partitions computer system resources into one or more partitions, wherein a first partition is associated with a first resource and a second partition is associated with a second resource. The system then allocates the first partition to a single resource pool, so that only processes associated with the single resource pool can access the first partition. At the same time, the system allocates the second partition to multiple resource pools so that processes associated with the multiple resource pools can share the second partition. In this way, non-critical resources can be shared, while other resources deemed critical to a workload's successful execution are not shared.

In one embodiment of the present invention, prior to allocating the different computer system resources, the system verifies that collective requirements of the one or more resource pools can be satisfied. If the collective requirements cannot be satisfied, the system signals an error condition.

In one embodiment of the present invention, establishing the first resource pool involves selecting a file containing a representation of the first resource pool from a number of possible files.

In one embodiment of the present invention, the system also stores a representation of the resource allocation to non-volatile storage so that the resource allocation can be reused after a machine failure. In a variation in this embodiment, the system stores a representation of each of the one or more resource pools along with associated resources. In a variation in this embodiment, the system stores an Extensible Markup Language (XML) representation of the resource allocation.

In one embodiment of the present invention, the first resource pool is associated with a first project, and the first process is one of a plurality of processes associated with the first project.

In one embodiment of the present invention, establishing the first resource pool involves establishing minimum and maximum requirements for a given resource.

In one embodiment of the present invention, the system dynamically adjusts the resource allocation during execution.

In one embodiment of the present invention, the different computer system resources can include, central processing units, semiconductor memory, swap space and networking resources.

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 illustrates a distributed computing system in accordance with an embodiment of the present invention.

FIG. 2 illustrates how computer system resources are allocated to resource pools in accordance with an embodiment of the present invention.

FIG. 3 illustrates the structure of a resource pool in accordance with an embodiment of the present invention.

FIG. 4 illustrates how processes are associated with projects in accordance with an embodiment of the present invention.

FIG. 5 is a flow chart illustrating the process of setting up a resource allocation in accordance with an embodiment of the present invention.

FIG. 6 is a flow chart illustrating the process of storing a resource allocation to a file in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

The data structures and code described in this detailed description are typically stored on a computer readable storage medium, which may be any device or medium that can store code and/or data for use by a computer system. This includes, but is not limited to, magnetic and optical storage devices such as disk

drives, magnetic tape, CDs (compact discs) and DVDs (digital versatile discs or digital video discs), and computer instruction signals embodied in a transmission medium (with or without a carrier wave upon which the signals are modulated). For example, the transmission medium may include a communications network, such as the Internet.

Distributed Computing System

FIG. 1 illustrates a distributed computing system 100 in accordance with an embodiment of the present invention. Distributed computing system 100 includes a collection of client computing systems 102-104 that are coupled to a server computing system 108 through a network 106.

Clients 102-104 can generally include any device on a network including computational capability and including a mechanism for communicating across the network. Server 108 can generally include any computing device including a mechanism for servicing requests from clients 102-104 for computational and/or data storage resources. Note that clients 102-104 and server 108 can generally include any type of computing device, including, but not limited to, a computer system based on a microprocessor, a mainframe computer, a digital signal processor, a portable computing device, a personal organizer, a device controller, and a computational engine within an appliance.

Clients 102-104 and server 108 include central processing units (CPUs) that execute threads. Threads are entities that generate a series of execution requests, while CPUs are entities that can satisfy the execution requests.

Network 106 can generally include any type of wire or wireless communication channel capable of coupling together computing nodes. This includes, but is not limited to, a local area network, a wide area network, or a

combination of networks. In one embodiment of the present invention, network 106 includes the Internet.

Server 108 includes an operating system 110 that supports flexible resource pools, which can be dynamically modified during system operation in accordance with an embodiment of the present invention.

Note that although the present invention is described in the context of a server computer system, the present invention is not limited to a server computer system. In general, the present invention can be applied to any computer system that allocates computational resources to different computational workloads.

Allocation of Resources to Pools

FIG. 2 illustrates how computer system resources are allocated to pools 220-222 in accordance with an embodiment of the present invention. As is illustrated in FIG. 2, server 108 contains various computational resources, including central processing units (CPUs) 202, memory 204, swap space 206, network interfaces 208 and scheduling classes 210. CPUs 202 include one or more CPUs within server 108. Note that it is possible to allocate an entire CPU to a pool, or alternatively, a fraction of a CPU. Memory 204 includes the main memory resources of server 108. Swap space 206 includes disk space that is used as a backing store for the virtual memory system of server 108. Network interfaces 208 include different channels for connecting server 108 with network 106. Note that network resources can alternatively be partitioned by allocating "available network bandwidth" instead of individual network interfaces.

Scheduling classes 210 are not actually system resources for which processes contend, but they can similarly be allocated to pools. For example, in FIG. 2, time-sharing scheduler 211 is assigned to pool 220, proportional share scheduler is assigned to pool 221 and real-time scheduler 212 is unassigned.

As is illustrated in FIG. 2, some of the resources within server 108 are allocated to pool 220, while other resources are allocated to pool 221. Note that both pool 220 and pool 221 share the same set of CPUs, while pools 220 and 221 do not share memory 204, swap space 206 or network interfaces 208. In this way, non-critical system resources can be shared, while other resources, deemed critical to a workload's successful execution, are not shared. This is an important advantage because sharing resources gives rise to more efficient resource utilization, which leads to better system performance.

As is illustrated in FIG. 2, server 108 also includes "pool free" 222 that contains system resources that are not assigned to specific pools.

Structure of Resource Pool

FIG. 3 illustrates the structure of resource pool 220 in accordance with an embodiment of the present invention. Resource pool 200 includes references to different resources, including a reference to a processor set 302, a reference to a memory set 304, a reference to a swap set 306, a reference to a network interface group 308 and a reference to a scheduling class 310. These references keep track of the associations between pool 220 and its resources. These references are indicated by arrows in FIG. 2.

Each of there references points to a resource data structure. For example, reference to processor set 302 points to resource data structure 320. Resource data structure 320 includes a number of items, including a list of the processor units 322 assigned to the resource. (Note that this list of units field only applies to certain resources, such as processors, that are allocated in discrete units. For other resources that are allocated as a range, such as memory, this field is not used.) Resource 320 also includes a minimum size 324 that needs to be allocated for the resource, as well as a maximum size 326 that needs to be allocated for the

resource. These minimum and maximum sizes are used by the system to automatically adjust the allocations assigned to resource 320 during system operation. Resource 320 also includes the actual size 328 of the allocation. Note that actual size 328 generally falls in the range from minimum size 324 to maximum size 326.

Note that pool 220 also includes properties 312 of the pool. For example, one of the properties can be an "importance" of the pool. If each of the pools includes an importance value, the system can use these importance values in adjusting the allocation of resources to pools. In doing so, the system can give preference to pools with higher importance values.

Processes and Projects

FIG. 4 illustrates how processes are associated with projects in accordance with an embodiment of the present invention. As is illustrated in FIG. 4, one or more tasks 403-404 are associated with a given project 402. Each of these tasks includes one or more processes. More specifically, task 403 includes processes 405 and 405, while task 404 includes process 407. Furthermore, each of these processes 405-407 can contain multiple threads (also referred to as light-weight processes). Note that when project 402 is associated with a resource pool, all of the associated processes 405-407 are associated with the same pool.

This project/task/process structure can be used to represent some types of applications. For example, project 402 may be associated with a database system, wherein tasks 403-404 are associated with specific components of the database system. Within the database system, processes 405-407 are dedicated to performing the actions involved in executing the specific components.

Process of Setting Up a Resource Allocation

FIG. 5 is a flow chart illustrating the process of setting up a resource allocation in accordance with an embodiment of the present invention. The system starts by selecting a file in non-volatile storage containing configuration information (step 502). Note that this file can be one of a number of different possible files containing resource allocation information for the computer system. In one embodiment of the present invention, configuration information within this file is encoded in extensible markup language (XML) format.

Next, the system opens the file (step 504) and then parses the configuration information to extract the configuration (step 506). For each resource requirement for a pool that is specified in the file, the system adds the minimum resource requirement to a collective requirement (step 508). The system then determines if the collective requirement is larger than the machine size (step 510). If so, the system signals an error condition and terminates because the collective requirement cannot be satisfied by the system (step 512).

Otherwise, if the collective requirement not larger than the machine size, the system construct partitions for each resource that is specified in the file (step 514). This can be accomplished by first meeting the minimum requirements for each pool, and then using a card dealing algorithm to dole out additional resources. If the system fails during this partitioning process, the system signals an error condition and terminates (step 512).

Once the resources are successfully partitioned, the system associates pools with the partitions (step 516). The system then binds each process to a specific pool that is associated with the process (step 518). In one embodiment of the present invention, this involves looking up a project that is associated with the process, and then looking up the pool that is associated with the project. Next, the system binds the process to each resource within the pool (step 520).

Note that this allocation of resources to processes is merely an initial allocation. This allocation can change over time as the system dynamically adjusts resource allocations based upon changing workload requirements (step 522).

Process of Storing a Resource Allocation

FIG. 6 is a flow chart illustrating the process of storing a resource allocation to a file in accordance with an embodiment of the present invention.

The system processes each resource in turn. For each resource, the system assigns a unique identifier to the resource (step 602), and then enumerates properties of the resource (step 604). Next, the system creates a resource node with properties as child nodes (step 606). The system then transforms this resource-property tree into an XML tag containing property sub-tags (step 608).

Next, the system processes each pool in turn. For each pool, the system identifies dependent resources by unique identifier (step 610). The system then transforms the pool, along with ID-based resource references into an XML pool tag containing references as attributes (step 612).

Next, the system commits the XML representation of the resources and the pools to a designated file (step 614). This allows the resources and pools to be reconstituted after a system failure.

Note that although the system described above uses an XML representation for resources and pools, in general any representation can be used. Hence, the present invention is not meant to be limited to an XML representation. After the configuration file is created, the system administrator can replicate the configuration file across multiple machines to guarantee a stable configuration. There may be some minor edits required (e.g. CPU names may differ, board

names may differ), but the configuration is essentially stable and very portable. Also, the configuration is persistent across reboots.

Furthermore, note that the configuration can be easily amended with small edits, such as altering the maximum amount of resource in a set, or major edits, such as the complete removal of all pools on the system.

Moreover, the above-described model is flexible. A system administrator may choose to bind multiple pools to a single resource, or may bind only one pool to a partition and thus provide guaranteed control of the partition for a pool. The system administrator may even leave a resource completely unutilized by associating no pools with the partition and leaving the resource as an "emergency standby partition".

With the above-described model, shifting workload is very easy. It simply involves associating the pool with a different set of resources. Furthermore, one or multiple resource sets may be changed and the resource sets can be changed many times over the lifetime of the pool.

Additionally, since the configuration document is XML, the configuration can be transformed into alternative formats easily, and can thus be re-used by XML-aware application which requires pool-related information. For instance, a pool monitoring application can read the dynamic XML configuration file and report the current configuration as an HTML document or a standard output text file.

Example Configuration File

The sample configuration file that appears below illustrates how resources and pools for a particular host can be represented in XML. Elements that contain other elements (for instance, processor_rset contains cpu) represent a containment relationship between those elements. Also, there are association relationships,

which represent relationships where elements require access to an uncontained element. For instance, pool elements have a resource_processor_rset attribute which references a defined processor_rset element.

```
<?xml version="1.0"?>
<!DOCTYPE pool_conf
    PUBLIC "-//Sun Microsystems Inc//DTD Resource Management All//EN"
    "file:///usr/share/lib/xml/dtd/rm_all.dtd">
<pool_conf>
    <processor_rset name="default" default ref_id="3452157">
        <cpu id="0" ref_id="2313243" />
        <cpu id="1" ref_id="7568334" />
        <cpu id="2" ref_id="6725923" />
        <cpu id="3" ref_id="4786376" />
    </processor_rset>
    <memory_rset name="default" default ref_id="7091674" unit="MB" size="2048" />
    <processor_rset name="small-0" id="0" ref_id="4845581">
        <cpu id="4" ref_id="5219421" />
        <cpu id="5" ref_id="6957092" />
        <cpu id="6" ref_id="7951354" />
        <cpu id="7" ref_id="3812561" />
    </processor_rset>
    <processor_rset name="small-1" id="1" ref_id="6520690"> <cpu id="8"  ref_id="7900695"
    />
        <cpu id="9"  ref_id="7716369" />
        <cpu id="10" ref_id="8321533" />
        <cpu id="11" ref_id="4773559" />
    </processor_rset>
    <processor_rset name="large-0" id="2" ref_id="6841430">
        <cpu id="12" ref_id="5596008" />
        <cpu id="13" ref_id="4675903" />
        <cpu id="14" ref_id="6997070" />
        <cpu id="15" ref_id="7944641" />
        <cpu id="16" ref_id="5091552" />
        <cpu id="17" ref_id="1401062" />
        <cpu id="18" ref_id="3872070" />
        <cpu id="19" ref_id="6022338" />
        </processor_rset> </processor_rset>
    <memory_rset name="medium-0" id="1" ref_id="8701782" unit="MB" size="1024" />
    <memory_rset name="medium-1" id="2" ref_id="1659240" unit="MB" size="1024" />
    <memory_rset name="small-0" id="3" ref_id="3981018" unit="MB" size="512" />
    <pool name="web_marketing" ref_id="3594665" resource_processor_rset="4845581"
    resource_memory_rset="8701782" importance="10" />
    <pool name="web_sales" ref_id="9338378" resource_processor_rset="6520690"
    resource_memory_rset="1659240" importance="10" />
    <pool name="app_marketing" ref_id="6784973" resource_processor_rset="6841430"
    resource_memory_rset="3981018" importance="20" />
```

</pool_conf>

The foregoing descriptions of embodiments of the present invention have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the present invention to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not intended to limit the present invention. The scope of the present invention is defined by the appended claims.